Support Vector Machines (SVM)

SVMs combine all the methods and ideas seen so far in order to find the best solution and also be robust to outliers.
Among all the linear models seen so far, this is the practical one.

(concept learning)

Let's consider a binary classification problem $y: X \longrightarrow \{+1, -1\}$ with dataset $D = \{(x_n, t_n)_{n=1}^N\}$, $t_n \in \{+1, -1\}$ and a linear model

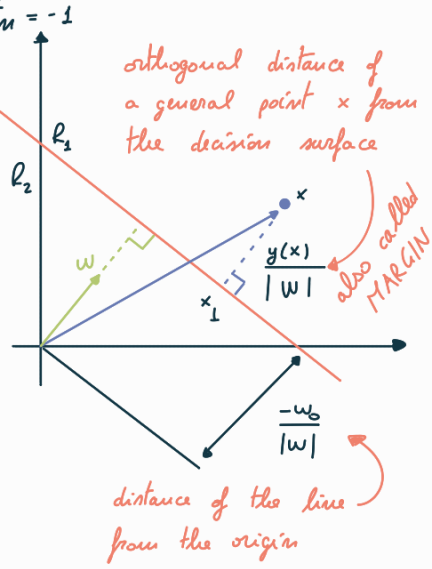
$$y(x) = w^T x + w_0$$

Assume D is linearly separable:

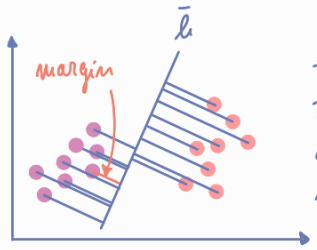
linearly separable: it exists an hyperplane characterized by w and w_0 that perfectly partitions the dataset.

$$\exists\, w, w_0 \mid y(x_n) > 0 \text{ if } t_n = +1, \quad y(x_n) < 0 \text{ if } t_n = -1$$

Given the geometric interpretation of a linear model, we want to maximize the distance between all the samples and the hyperplane.
MARGIN: smallest distance between a sample and the decision boundary.
In SVMs the decision boundary is chosen to be the one for which the margin is maximized.



$$\text{margin} = \min_{n=1,\dots,N} \frac{|y(x_n)|}{|w|} = \frac{1}{|w|} \min_{n=1,\dots,N} |w^T x_n + w_0|$$



the margin is the minimum among the distances of all the samples in the dataset with respect to a particular hyperplane

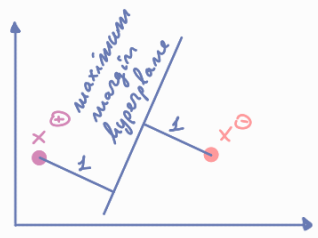
$$\bar{h} = \bar{w}^T x + \bar{w}_0$$

What we want: $w^*, w_0^* = \underset{w, w_0}{\text{argmax}} \frac{1}{|w|} \min_{n=1,\dots,N} |w^T x + w_0|$ / w, w_0 that maximize the margin

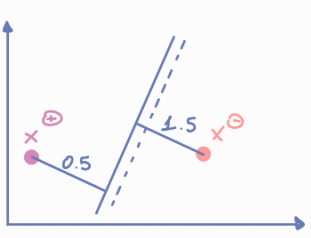
Direct optimization problem would be very complex to solve but we can convert it into an equivalent problem much easier to solve:

Rescale the dataset such that for the closest point x_k we have $t_k(w^T x_k + w_0) = 1$
All the other points will have $t_n(w^T x_n + w_0) \geq 1 \quad (\forall n = 1, \dots, N)$.

When we find the optimal hyperplane, there will exist at least one point for each class such that its distance is exactly 1.



correct ← we set min distance = 1



it cannot be other than that otherwise the chosen $\bar{h}$ won't be the best hyperplane

We will use only these points and discard all the others.

from this formulation we can move to the lagrangian multipliers

$$w^*, w_0^* = \underset{w}{\text{argmax}} \frac{1}{|w|} = \underset{w}{\text{argmin}} \frac{1}{2}|w|^2$$

This is an example of a quadratic programming problem in which we are trying to minimize a quadratic function subject to a set of linear inequality constraints.

Quadratic programming problems can be solved with the lagrangian method. We obtain:

$$w^* = \sum_{n=1}^{N} \alpha_n^* t_n x_n$$

dimension of the input space ~ number of features

α_i^* = lagrange multipliers

Optimization problem for determining w (dimension $|X|$) transformed in an optimization problem for determining α (dimension $|D|$) → efficient when $|X| \ll |D|$

dimension of the dataset (one lagrange multiplier for each sample)

Why are we transforming an optimization problem in few dimensions (e.g. 2) into an optimization problem in thousand dimensions?

↳ when the value of α_i^* is 0 then the corresponding sample does not contribute to the solution.

& This optimization problem satisfies the KKT condition:

Karush-Kuhn-Tucker (KKT) condition:

for each $x_n \in D$, either $\alpha_n^* = 0$ or $t_n y(x_n) = 1$, thus $\boxed{t_n y(x_n) > 1}$ implies $\alpha_n^* = 0$

if $t_n y(x_n) = 1$ then $\alpha_n^* \neq 0$

all the points that are not the closest one

The points that are not the closest ones (one for each class) does not contribute

Support vectors: x_k such that $t_k y(x_k) = 1$ and $a_k^* > 0$

$$SV \equiv \{ x_k \in D \mid t_k y(x_k) = 1 \}$$

The solution only depends on the samples that are in this set

$$y(x) = \sum_{x_j \in SV} \alpha_j^* t_j x^T x_j + w_0^* = 0$$

$$y(\tilde{x}) = w^{*T} \tilde{x} + w_0$$

$$w^* = \sum_{x_j \in SV} \alpha_j t_j x_j$$

Other vectors $x_n \notin SV$ do not contribute ($\alpha_n^* = 0$)

$$y(\tilde{x}) = w_0 + \sum_{x_j \in SV} \alpha_j t_j \tilde{x}^T x_j$$

To compute w_0:

① Support vector $x_n \in SV$ satisfies $t_k y(x_k) = 1$

$$t_k \left(\sum_{x_j \in SV} \alpha_j^* t_j x_k^T x_j + w_0^* \right) = 1$$

② Multiply by t_k and using $t_k^2 = 1$:

$$\cancel{t_k^2}^{1} \left(\sum_{x_j \in SV} \alpha_j^* t_j x_k^T x_j + w_0^* \right) = t_k \quad (y(x_k))$$

$$w_0^* = t_k - \sum_{x_j \in SV} \alpha_j^* t_j x_k^T x_j$$

$$y(x) = w^T x + w_0 \rightarrow w_0 = y(x) - w^T x$$

Instead of using one particular support vector x_n to determine w_0, a more stable solution is obtained by averaging over all the support vectors

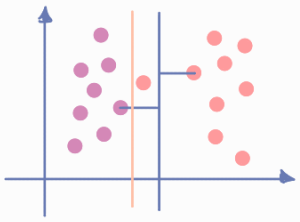
$$w_0^* = \frac{1}{|SV|} \sum_{x_k \in SV} \left(t_k - \sum_{x_j \in SV} \alpha_j^* t_j x_k^T x_j \right)$$

Given the maximum margin hyperplane determined by α_m^*, w_0^* (all the Lagrange multipliers), to classify a new instance:

$$\text{sign}(y(x')) = \text{sign}\left(\sum_{x_k \in SV} \alpha_k^* t_k x'^T x_k + w_0^*\right)$$

we only need to determine the lagrange multipliers for the support vectors.

What if the data is almost linearly separable (e.g. few data points are on the other side of the true hyperplane)?



- hyperplane that the SVM we have seen so far would have found (worse)
- true hyperplane robust to noise

Even if ■ is the optimal solution for the actual dataset (accounting for noise) we still have that ■ will be better when we have to classify a new instance

How to deal with this? How to extend the problem in such a way that it can cope with this situation?

The constraint $t_m(w^T x_m + w_0) \geq 1$ means that all the points must be outside of the margin.

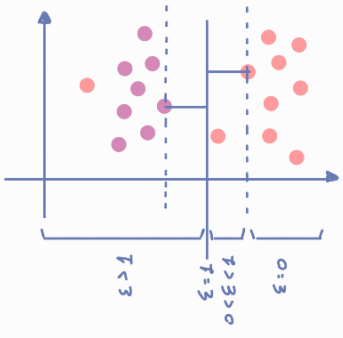
$t_m \quad y(x_m) \quad \geq 1$

We want to relax this constraint to allow some points to be inside the margin. We will introduce some variables in the formulation of the problem that will measure how much the constraint is violated: these variables will be called SLACK variables.

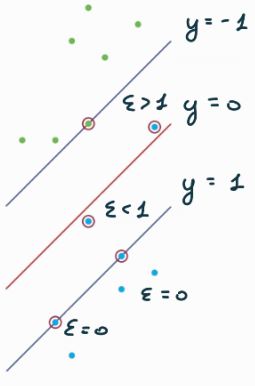
$$\varepsilon_n \geq 0 \qquad n = 1, \dots, N$$

The new, less strict constraint will be called soft margin constraint.

$\varepsilon_n = 0$	if the point is inside the correct margin boundary
$0 < \varepsilon_n < 1$	if the point is inside the margin but still on the right side with respect to the optimal solution
$\varepsilon_n = 1$	if the point is exactly on the optimal plane
$\varepsilon_n > 1$	if the point is beyond the optimal plane



$\varepsilon > 1$ means that the sample will be missclassified by the solution



How to soften the constraint?

$$t_m y(x_m) \geq 1 \longrightarrow t_m y(x_m) \geq 1 - \varepsilon_m$$

We also want to minimize all the ε_n

$$w^*, w_0^* = \text{argmin}\left(\frac{1}{2}|w|^2 + C\sum_{n=1}^{N} \varepsilon_n\right)$$

first term to minimize is the maximum margin; second term to minimize is the sum of the slack variables

C is a factor that weights how much you want to penalize the slack variables with respect to the maximization of the margin

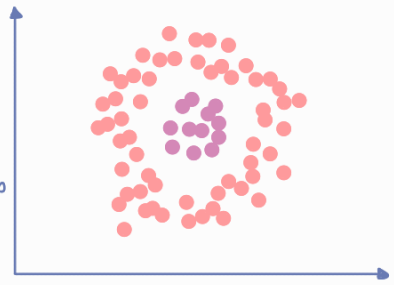
non-linearly separable data due to noise

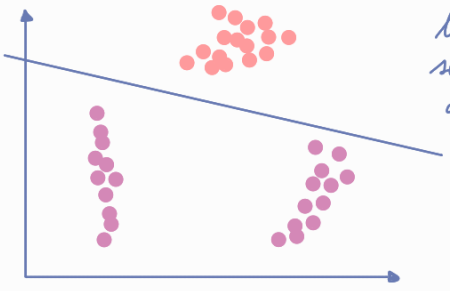
The solution of this optimization problem is similar to the previous one (this is still a quadratic programming problem that can be solved with the lagrangian optimization

$$w^* = \sum_{n=1}^{N} \alpha_n^* t_n x_n \quad ; \quad w_0^* = \frac{1}{|SV|} \sum_{x_k \in SV} \left(t_k - \sum_{x_j \in SV} \alpha_j^* t_j \, x_k^T x_j \right)$$

This model is robust to outliers and to noise.
How to deal with non-linearly separable data due to the distribution itself (not to noise)?

intrinsically non-linearly separable data. We can make a change of coordinates (cartesian → polar) and consider radius as a feature





linearly separable data

So far we considered models working directly on x. All the results hold if we consider a non-linear transformation of the inputs $\phi(x)$ called basis function. Decision boundaries will be linear in the feature space ϕ and non-linear in the original space x.
Classes that are linearly separable in the feature space ϕ may not be separable in the input space x.

$$D = \{ (x_n, t_n) \} \quad x_n \in \mathbb{R}^d \longrightarrow D_\phi = \{ (\phi_n = \phi(x_n), t_n) \}$$

$$\phi_n = \phi(x_n) = \begin{bmatrix} \phi_0(x) \\ \vdots \\ \phi_m(x) \end{bmatrix} \quad m \neq d$$

$\phi_0(x)$ fixed to 1

We don't need to invert ϕ

SVM ← linearly separable data
almost linearly separable data
non-linearly separable data

Support Vector Machines (SVM)

$x \in \mathbb{R}^D$

We have a set of points we want to classify.
Each point is represented by a feature vector $x \in \mathbb{R}^D$

$\phi : \mathbb{R}^D \longrightarrow \mathbb{R}^M$
$\phi(x) \in \mathbb{R}^M$

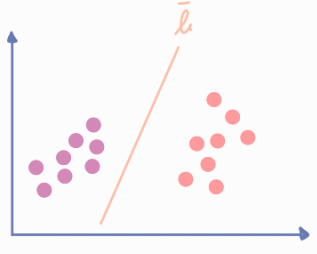
basis vector

This may be too simple for many applications, so we want to map this to a more complex, non-linear feature space ϕ.

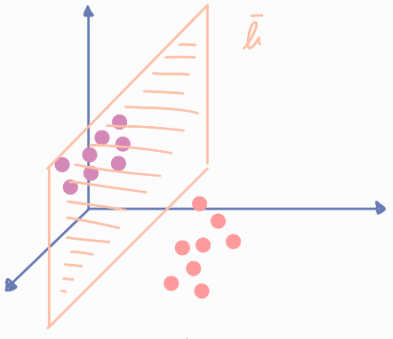
A decision boundary is a separator that divides these points into their respective classes. This hyperplane is represented as:

$$H : w^T \phi(x) + w_0 = 0$$

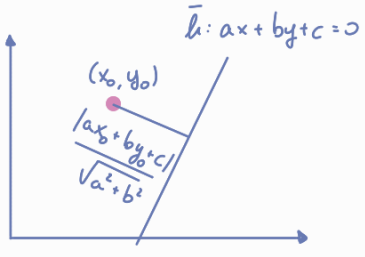
In a d-dimensional space an hyperplane is a $d-1$-dimensional structure



in a 2-dimensional space the hyperplane is a 1-dimension line

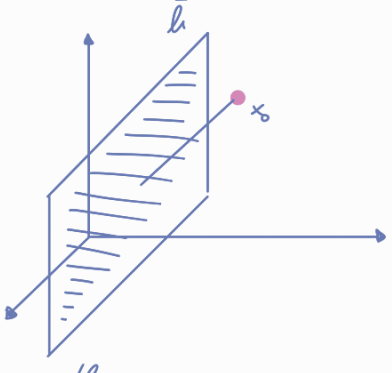


in a 3-dimensional space the hyperplane is a 2-dimension plane



the distance of a point from a line is this:

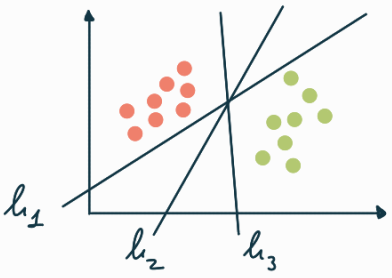
$$\frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$



in the same way we can express the distance of a point from an hyperplane:

$$d_H(\phi(x_0)) = \frac{|w^T \phi(x_0) + w_0|}{\|w\|}$$

Consider the case where the data is perfectly separable. In this case it exists one hyperplane that can separate the training data groups with 100% accuracy.
Consider a concept learning problem (classification of two classes):

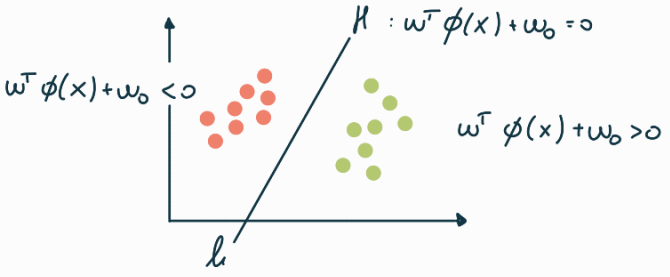


how do we know which hyperplane to choose?

Intuitively we can choose the hyperplane for which the distance from the closest points is maximum. This way we will make less mistakes during testing.

$$w_* = \underset{w}{\operatorname{argmax}} \left(\min_n d_H(\phi(x_n)) \right)$$

The goal of SVM is defining the hyperplane that has the maximum margin from the closest points (in other words: maximize the minimum distance).



Stick with the concept learning problem for now:

Use this convention: $y_n \begin{cases} +1 & \text{for } \bullet \\ -1 & \text{for } \bullet \end{cases}$

This way: $y_n(w^T\phi(x_n) + w_0) \begin{cases} \geq 0 & \text{correct} \\ < 0 & \text{incorrect} \end{cases}$

ground truth y_n	prediction $(w^T\phi(x_n) + w_0)$	$y_n(w^T\phi(x_n) + w_0) = 1$
>0	>0	>0
<0	<0	>0

① Perfect classification

$$w^* = \underset{w}{\operatorname{argmax}} \left(\min_n d_H(\phi(x_n)) \right) = \underset{w}{\operatorname{argmax}} \left(\min_n \frac{|w^T\phi(x_n) + w_0|}{\|w\|} \right)$$ } what we had so far

$$= \underset{w}{\operatorname{argmax}} \left(\min_n \frac{y_n(w^T\phi(x_n) + w_0)}{\|w\|} \right)$$ } equal to the modulo if all the samples are correctly classified } whith the new convention

$$= \underset{w}{\operatorname{argmax}} \frac{1}{\|w\|} \left(\min_n y_n(w^T\phi(x_n) + w_0) \right)$$ represents the distance of the closest point to H

Let $\min_n y_n(w^T\phi(x_n) + w_0) = 1$; with this normalization we impose that the distance of the closest point must be 1

this is possible by multiplying some constant factor c to the weight vector w and to the bias w_0 $\quad w \leftarrow cw$, $w_0 \leftarrow cw_0$

We obtain:

$$w^* = \underset{w}{\operatorname{argmax}} \frac{1}{\|w\|}$$

From this formulation we can move to an equivalent one:

$$w^* = \arg\max \frac{1}{\|w\|} = \arg\min \frac{1}{2}\|w\|^2$$

This is the primal formulation of the SVM for perfectly separable data and assuming that all the points are classified correctly; in particular the latter assumption is often violated in reality since also the training set is subject to noise.

This is an example of a quadratic programming problem in which we are trying to minimize a quadratic function subject to a set of linear inequality constraints.

② Non-perfect classification

Instead of trying to classify every single point correctly and risk overfitting we allow to make some mistakes by using slack variables.

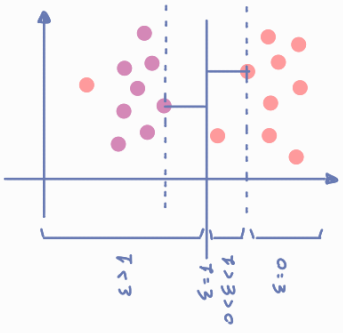
~~$y_n(w^T\phi(x_n)+w_0) > 0 \quad \forall n$~~

$$y_n(w^T\phi(x_n)+w_0) \leq 0 \quad \exists n$$

New primal form of SVM:

$$\min_{w, w_0, \{\xi_n\}} \frac{1}{2}\|w\|^2 + C\sum_n \xi_n$$

$$y_n(w^T\phi(x_n)+w_0) \geq 1-\xi_n \quad \forall n \qquad \xi_n \geq 0 \quad \forall n$$ constraints



$\xi > 1$ means that the sample will be misclassified by the solution

Low C ⟶ less complex boundary
High C ⟶ more complex boundary

These (case ① and ②) are examples of convex quadratic optimization problems: the objective function is quadratic in w and the constraints are linear in w and ξ.
To solve these constraint optimization problems we use the method of Lagrange multipliers.

Lagrange Multipliers

① Obtain the primal form

$$\min_x f(x)$$

$$g_i(x) \leq 0$$

② Derive the lagrangian function

$$L(x, \{\lambda_i\}) = f(x) + \sum_{i=1}^{n} \lambda_i g_i(x)$$

$\lambda_i \geq 0$ ⟶ lagrange multipliers

③ Solve $\partial L / \partial x = 0$

④ We have a set of

$x = h(\{\lambda_i\})$

(isolate the original variables)

⑤ Substitute back in the Lagrangian

$$L(\{\lambda_i\}) = f(h(\{\lambda_i\})) + \sum_{i=1}^{M} \lambda_i g_i(h(\{\lambda_i\}))$$

We must have a function of only dual variables

⑥ Rewrite the problem

$$\max_{\{\lambda_i\} \geq 0} \min_{x} L(x, \{\lambda_i\})$$

Applied to our case we have:

① $$\min_{w, w_0, \{\varepsilon_n\}} \frac{1}{2}\|w\|^2 + C\sum_n \varepsilon_n$$ } target function

$y_n(w^T\phi(x_n) + w_0) \geq 1 - \varepsilon_n \quad \forall n$

$\varepsilon_n \geq 0 \quad \forall n$ } constraints

① $\longrightarrow$ $1 - \varepsilon_n - y_n w^T \phi(x_n) - y_n w_0 \leq 0$

② $\longrightarrow$ $-\varepsilon_n \leq 0$

② Determine the Lagrangian

$\{\varepsilon_n\}$ → set of slack variables, one for each sample

$\{\lambda_n\}$ → set of lagrange multipliers, one for each sample for constraint ①

$\{\alpha_n\}$ → set of lagrange multipliers, one for each sample for constraint ②

$$L(w, w_0, \{\varepsilon_n\}, \{\lambda_n\}, \{\alpha_n\}) = f(h(\{\lambda_i\})) + \sum_{i=1}^{M} \lambda_i g_i(h(\{\lambda_i\}))$$

$$= \frac{1}{2}\|w\|^2 + C\sum_n \varepsilon_n + \sum_n \alpha_n (1 - \varepsilon_n - y_n(w^T\phi(x_n) + w_0)) + \sum_n \lambda_n(-\varepsilon_n) \quad \forall n$$

① ②

③ Solve $\partial L / \partial x = 0$ (original variables are $w, w_0, \{\varepsilon_n\}$

$$L(\cdot) = \frac{1}{2}\|w\|^2 + C\sum_n \varepsilon_n + \sum_n \alpha_n - \varepsilon_n \alpha_n - \alpha_n y_n w^T \phi(x_n) - \alpha_n y_n w_0 - \sum \lambda_n \varepsilon_n$$

$\partial L / \partial w = w - \sum \alpha_n y_n \phi(x_n) = 0$ (1/2 ✓ w)

$\partial L / \partial w_0 = \sum_n \alpha_n y_n = 0$

$\partial L / \partial \varepsilon_n = C - \alpha_n - \lambda_n = 0$

$w - \sum_n \alpha_n y_n \phi(x_n) = 0$

$w = \sum_n \alpha_n y_n \phi(x_n)$

recall those two rules for vector differentiation:

① $\frac{\partial(x^T a)}{\partial x} = \frac{\partial(a^T x)}{\partial x} = a^T$ with $x, a \in \mathbb{R}^{M \times 1}$

note also that $x^T a = a^T x$

② derivative of the transpose of a vector is the same as the derivative of the vector

⑤ Substitute these values back in the Lagrangian to eliminate the primal variables

$$L(\cdot) = \frac{1}{2}\left(\sum_m \alpha_m y_m \phi(x_m)\right)^T \left(\sum_n \alpha_n y_n \phi(x_n)\right) + C\sum_n \varepsilon_n + \sum_n \left(\alpha_n \left(1 - \varepsilon_n - y_n \left(\sum_m \alpha_m y_m \phi(x_m)\right)^T \phi(x_n)\right.\right.$$

$\frac{1}{2}$; $\|w\|^2$

$$+ C\sum_n \varepsilon_n + \sum_n \left(\alpha_n \left(1 - \varepsilon_n - y_n \left(w^T \phi(x_n)\right.\right.\right.$$

$$\left.\left.\left. + w_0\right)\right)\right) + \sum_n \lambda_n(-\varepsilon_n)$$

$$\left.\left. + w_0\right)\right)\right) + \sum_n \lambda_n(-\varepsilon_n)$$

■ old formula

■ new formula

$$L(\cdot) = \frac{1}{2}\sum_m \sum_n \alpha_m \alpha_n y_m y_n \phi^T(x_m)\phi(x_n) + C\sum_n \varepsilon_n + \sum_n \alpha_n - \sum_n \alpha_n \varepsilon_n$$
$$- \sum_m \sum_n \alpha_m \alpha_n y_m y_n \phi^T(x_m)\phi(x_n) - w_0 \sum_n \alpha_n y_n - \sum_n \lambda_n \varepsilon_n$$
$$= -\frac{1}{2}\sum_m \sum_n \alpha_m \alpha_n y_m y_n \phi^T(x_m)\phi(x_n) + \dots$$
$$\dots = \sum_n \alpha_n - \frac{1}{2}\sum_m \sum_n \alpha_m \alpha_n y_m y_n \boxed{\phi^T(x_m)\phi(x_n)}$$

this is the important part

This is the optimization problem that we want to solve:

$$\begin{cases} \max_{\alpha} \quad \Theta(\alpha) = \sum_m \sum_n \alpha_m \alpha_n y_m y_n \phi^T(x_m)\phi(x_n) \\ 0 \le \alpha_i \le C \\ \sum_m \alpha_m y_m = 0 \end{cases}$$

This can be a very large QP optimization problem. We can use various techniques to tackle it but the best one is the Sequential Minimal Optimization Algorithm.
I won't go into details.
The important part is that we have $\phi^T(x_m)\phi(x_n)$ and we will see that with the kernelization we can compute it disregarding the true form of $\phi(\cdot)$